OIPE
AUG 1 8 2003
PATENT & TRADEMARK OFFICE

500.42884X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): T. ENDO, et al

Serial No.: 10/608,209

Filed: June 30, 2003

Title: INFORMATION PROCESSING MEANS

RECEIVED
MAR 0 3 2004
Technology Center 2100

LETTER CLAIMING RIGHT OF PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

August 18, 2003

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on:

Japanese Patent Application No. 2003-014136
Filed: January 23, 2003

A certified copy of said Japanese Patent Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/rp
Attachment

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００３年　１月２３日

出願番号
Application Number: 特願２００３－０１４１３６

[ST.10/C]: ［ＪＰ２００３－０１４１３６］

出願人
Applicant(s): 株式会社日立製作所

２００３年　６月１０日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０４４９７５

【書類名】　特許願
【整理番号】　H02016081A
【あて先】　特許庁長官　殿
【国際特許分類】　H04L　9/10
【発明者】
　【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　遠藤　隆
【発明者】
　【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　神永　正博
【発明者】
　【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
　【氏名】　渡邊　高志
【特許出願人】
　【識別番号】　000005108
　【氏名又は名称】　株式会社　日立製作所
【代理人】
　【識別番号】　100075096
　【弁理士】
　【氏名又は名称】　作田　康夫
　【電話番号】　03-3212-1111
【手数料の表示】
　【予納台帳番号】　013088
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　情報処理方法

【特許請求の範囲】

【請求項1】

＾をベキ乗演算と定義し、Pを素数とし、x＞Pであるxに対して、x＊（2＾n）　mod P を計算する情報処理方法に於いて、法をP、入力値をx、法Pを格納するために必要充分なビット数をn、入力値xを格納するために必要なビット長をmとし、入力値をx＊（2＾n）　mod Pに変換する際に、2＾(2m+n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算によりx1＝x＊2＾（2m＋n）＊（2＾（－m））mod P＝x＊2＾（m＋n）　mod Pを計算し、さらにx2：＝x1＊（2＾（－m））mod P＝x＊（2＾n）　mod Pを計算することにより、x　mod Pを陽に求めること無しにx＊（2＾n）　mod P を計算することを特徴とする情報処理方法。

【請求項2】

＾をベキ乗演算と定義し、Pを素数とし、x＞Pであるxに対して、x＊（2＾n）　mod P を計算する情報処理方法に於いて、法をP、入力値をx、法Pを格納するために必要充分なビット数をn、入力値xを格納するために必要なビット長をmとし、入力値をx＊（2＾n）　mod Pに変換する際に、2＾(m+2n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算方式によりx1＝x＊2＾（m＋2n）＊（2＾（－m））mod P＝x＊2＾（2n）　mod Pを計算し、さらにx2：＝x1＊（2＾（－n））mod P＝x＊（2＾n）　mod Pを計算することにより、x　mod Pを陽に求めること無しにx＊（2＾n）　mod P を計算することを特徴とする情報処理方法。

【請求項3】

法をP、入力値をx、法Pを格納するために必要充分なビット数をn、入力値xを格納するために必要なビット長をm、指数をdとし、ベキ乗の指数をsビットづつとりだしてsビット毎のベキ乗演算の結果を合成して、x＾d　mod Pのベキ乗剰余演算を行う情報処理方法において、sビットづつ取り出した指数のi番目の指数をd[i]としたとき、x＾d[i] modPを演算する代わりに（2＾n

）＾（２＾ｓ－１）＊ｘ＾d[i]mod　Ｐを用いて演算を行い、（２＾ｎ）＾（２＾ｎ－１）＊ｘ＾ｄ　ｍｏｄ　Ｐを計算した後、２＾（－ｎ）＾（２＾ｎ－１）ｍｏｄ　Ｐを乗じて、ｘ＾ｄ　ｍｏｄ　Ｐを計算することを特徴とする情報処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は情報処理方法に関し、特に例えば、機密性の高いＩＣカードなどの耐タンパ装置に関するものである。

【０００２】

【従来の技術】

RSAの高速演算手法のCRT(Chinese Remainder Theory)演算方式では、計算の一番最初のステップでx mod pを計算する。ＣＲＴ演算方式の処理手順を図１に示す。まず、ｐで還元した値に対して剰余べき乗演算を行い（１０１０）、ｑで還元した値に対するべき乗剰余演算を行う（１０２０）。最後に２つのべき乗剰余演算結果を合成し（１０３０）最終的な結果を得る。最初のｐで還元した値の剰余べき乗演算（１０１０）およびｑで還元した値のべき乗剰余演算（１０２０）の最初のステップで、それぞれ秘密指数ｐに対する剰余、秘密指数ｑに対する剰余を計算する必要がある。べき乗剰余を計算するには、剰余乗算を繰り返し行うことで実現する。

【０００３】

１０１０、１０２０のべき乗計算には通常、アディションチェイン手法を用いる。アディションチェインとは、たとえば、Z=A^L を計算する場合、指数Ｌを二進数展開し、

L=L[n-1]*2^(n-1)+L[n-2]*2^(n-2)+ … +L[1]*2^1+L[0]*2^0　　（式１）

と置き、指数の加算は掛算になり、指数の乗算がベキ計算になるという指数法則を用いて、Ｚ＝Ａ＾Ｌの計算を

Z := (…(((A^L[n-1])^2 * (A^L[n-2]))^2 * … * (A^L[0])　　（式２）

とおく。A^L[i]は、L[i]=1であれば、Aとなり、L[i]=0 であれば、A^0=1 となり

、L[i]=0の場合に１の掛算を省略すると、Lを２進表記した際の１となるビット数回の掛算と、m-1回の自乗計算により、A^Lが計算できる。

プログラムにより表現すると、

```
Z= 1
for ( i = n-1 ; i >= 0 ; i++ ) {
  W := W * W;
  if ( L[i]==1 ) then Z:=Z * A; else W:=W * 1;
}
```

となる。

【０００４】

剰余乗算の方式には大きくモンゴメリ剰余乗算を用いたものと、そうでないものの２つに分けることができる。

【０００５】

図２はモンゴメリ剰余乗算を用いた場合のアディションチェインによるべき乗剰余計算の処理フローである。nは、Ｐを格納するのに十分なビット長を示す。まず、Ｐに対するｘの剰余を２０２０で求める。モンゴメリ剰余乗算では、乗算の度に2^(-n) mod P が乗ぜられるため、予め2^nを被演算数に乗じておく。以下、Ｒ＝2^nとおく。被演算数にＲを乗ずるのにも、モンゴメリ剰余乗算を用いる。あらかじめR^2を計算しておき、モンゴメリ剰余乗算を用いてx mod Pに乗じ、xR mod P を得る（２０４０）。モンゴメリ剰余乗算を用いるので、演算の初期値は１の代わりに１にＲを乗じたＲとする。乗算の処理は、指数の最上位のビットから１ビットづつ取り出すので、カウンタiに最上位ビットの位置を示すn-1を設定する（２０５０）。指数上位ビットから順に１が立っているかどうかを調べ（２０６０）、０であれば、１を意味するＲを乗じ（２０７０）、１が立っていれば、xR mod P(=AR)をWに乗じる（２０８０）。１を意味するＲを乗じる処理２０７０は、計算結果に影響を与えないので、処理速度を重視する場合は省略可能である。ビットの位置カウンタを１ビット分移動し（２０９０）、最下位ビットまでたどり着いているかチェックを行い（２１００）、最下位ビットまで達していない場合は結果を二乗し（２１１０）、指数の次のビットに対して３０６０か

らの処理から処理を繰り返す。２１００で最下位ビットまで処理が終了した場合は、2^nが掛かっている影響を取り除くため、2^(-n)を乗じる。モンゴメリ剰余乗算で、１との積を計算することは、2^(-n)を乗じることと等しい（３１２０）。最後に、結果がＰ以上の場合は（２１３０）、Ｐを減ずる（２１４０）。一連の処理の中で、２０２０の剰余演算の結果は、図3に示すように、Ｐの倍数（３０１０）を境にＰより大であるか小であるかによって、大きく値が変動するため、アタックポイントとなる可能性がある。

【０００６】

「図４」はモンゴメリではない剰余乗算を用いてアディションチェインによるべき乗剰余演算を行った場合の処理フローである。Ｐのビット長をｎとおく。つぎに、まずｘのＰに対する剰余を４０２０で求める。モンゴメリ剰余乗算を用いた場合の処理と同様、このＰに対する剰余を求める演算がアタックポイントとなる可能性がある。通常の剰余乗算を用いるので、演算の初期値は１とし、最上位のビットから１ビットづつ取り出すので、カウンタiに最上位ビットの位置を示すn-1を設定する（４０４０）。指数上位ビットから順に１が立っているかどうかを調べ（４０５０）、１が立っていれば、x mod P(=A) をWに乗じ（４０７０）、ビットの値が０であれば、１を意味を乗じる（４０６０）。また、１を乗じる処理４０６０は、計算結果に影響を与えないので、処理速度を重視する場合は省略可能である。４０７０の処理でx mod P の値が被演算数に使用されるので、ここもアタック対象になる可能性がある。ビットの位置を１ビット下に移動し（３０８０）、最下位ビットまでたどり着いていない場合は、結果を二乗し（３０９０）、指数の次のビットの処理を行う。最下位ビットまで処理が終了した場合は、その時点でのWが計算結果となる。

【０００７】

以上のように、モンゴメリ剰余乗算を用いている場合も、通常の剰余乗算を行った場合も、Ｐによる剰余算結果が処理の最初に必要となりアタックポイントとなる可能性がある。

【０００８】

【発明が解決しようとしている課題】

RSA暗号は認証や、秘密鍵の配送などに標準的に用いられている暗号で、金融用途等ではその演算の安全性が非常に重要視されている。RSA暗号の高速演算手法として、中国人剰余定理を用いた計算手法が広く用いられているが、その一番最初の演算で、秘密素数pによる剰余計算が必要となる。この計算は、秘密素数pを陽に用いた計算であるので、古くからアタックの対象となっている。pによる剰余計算で問題となるのは、図２に示すようにx がpの倍数近傍（２０１０）の値の場合 x<kp では、x mod p≒p となり大きな値となる一方、x>kpの場合は、x mod p≒0となり、小さな値となることである。x mod p の値がpを境界として大きく変動するため、入力xが秘密指数pよりも大なのか小であるのかが、電流値等のサイドチャネル情報として識別できる危険性がある。RSA暗号では、大きな素数(現在は、512bit程度の素数が用いられている)p、 qの積Nが容易に因数分解できないことを安全性の根拠としており、Nは公開鍵の一部としてユーザに公開されている。秘密素数pもしくはqがリークすると、N/pは容易に計算可能であるので、秘密鍵dは公開鍵eの(p-1)(q-1)を法とする逆元を計算することにより、求めることが出来る。逆元計算は、拡張ユークリッド互除法により、容易に計算可能である。本発明は、ＣＲＴ向けの剰余計算を高速かつ安全に行うための計算方法および装置に関するものである。

【０００９】

【課題を解決するための手段】

x mod p を直接計算せずに、2^(m+n) mod p もしくは、2^(2n) mod pをあらかじめxに乗じておき、その後、2^(-n)もしくは、2^(-m)を乗じて、2^n x mod p を計算する。pは大きな素数であるので常に奇数となり、2のべき乗とは常に互いに素になるので、2^(-m)mod p もしくは 2^(-n) mod p は必ず存在する。また、2^n x mod p の値は、入力xがpの近傍にある場合でもxとpの大小関係に依存した値の大きな変化は発生せず、2^m x mod pのビット長はpのビット長に近い値となる。したがって、リーク情報からxとpの大小関係を推定することが不可能となり、秘密鍵の漏洩を防止することを可能とする。モンゴメリ剰余乗算を用いる場合には、2^n を乗じた形式は、そのままモンゴメリフォーマットになっているので、以降の処理は、従来の処理フローを使用できる。

【0010】

モンゴメリ剰余乗算を用いない場合には、指数演算計算の最後に、(2^(-n))^(2^n-1) mod p を乗じ、2^n mod p を乗じた影響を補正することで、正しい結果を得る。

【0011】

モンゴメリ乗算以外の剰余乗算を用いる場合は、乗算と自乗を行った後、R^(-2) mod pを乗じる。あらかじめ R^(-2m) mod p を計算しておき、最後にR^(-(2^m)+1) mod p を乗じて補正してもよい。

【0012】

【発明の実施の形態】

図５はモンゴメリ剰余乗算を用いた場合の本発明の一実施例を示す。ｍを入力ｘの格納に必要なビット長、ｎをＰの格納に必要なビット長とする。０≦ｘ≦Ｐ＊Ｑであるので、必ずｍ≧ｎとなる。まず、U=2^U_SQR=2^(2n)U mod Pを計算する（５０３０）。U_SQR=2^(2n)U mod Pを計算する部分の詳細な処理フローを図７に示す。図７では、2^L*U mod P の計算手順が示されているが、L=2nとして、図７の処理を用いることができる。U_SQRのビット長は、m-2nもしくはnのうちの長い方のビット長になる。５０４０の計算は、

A_R = (x * U_SQR + M*p) / 2^m　　　　（式３）

となり、x<2^m、M<2^mであるので、

A_R < U_SQR + p　　　　（式４）

となる。pのビット長はn以下であるので、A_Rのビット長は、MAX(m-2n、 n)となる。このビット長がn以下となるには、

m-2n < n　　　　（式５）

m < 3n　　　　（式６）

である必要がある。通常の場合、m≒2n<3n であるので、A_Rのビット長はnとなる。５０５０の処理を行うには、５０４０の処理結果がn以下となる必要がある。m < 3n の条件を満たさない場合には、図６に示す別の実施例による方法を行う。５０５０の処理を別の式で書くと、

$(A\_R + (-A\_R*p^(-1) \bmod 2^n) * p) / 2^n$ （式7）

であるが、$A\_R < 2^n$ かつ $(-A\_R * p^(-1) \bmod 2^n) < 2^n$ であるので、

$(A\_R + (-A\_R*p^(-1) \bmod 2^n) * p) / 2^n < 1+p$ （式8）

となり、必ず５０５０の処理の後の A_R は p以下となり、nビット長で表現できる。また、５０５０でA_Rの値がpと等しくなるのは、xの値がpの倍数となる場合のみである。

５０３０から５０５０までの処理を数式で表現すると、

$A\_R \equiv x * 2^(2n) * 2^(m) * 2^(-m) * 2(-n) \bmod P$ （式９）

$\equiv x * 2^(2n+m-m-n) \bmod P$ （式１０）

$\equiv x * 2^n \bmod P$ （式１１）

となる。

５０５０以降の処理は、図３の３０５０以降の処理と同一である。xの値がｐの倍数となっている場合、図２中の２１３０、２１４０の処理にて最終的に補正される。

【００１３】

図６はモンゴメリ剰余乗算を用いた場合の本発明の別の一実施例を示す。ｍを入力ｘの格納に必要なビット長、ｎをＰの格納に必要なビット長とする。０≦ｘ≦Ｐ＊Ｑであるので、必ずｍ≧ｎとなる。まず、L=n+mとして、図７に示されるフローに従って、U'_SQR=2^(n+m)U mod Pを計算する（６０３０）。６０４０以降の処理を行うためにはU'_SQRのビット数はm以下である必要があるが、U'_SQRは必ずmビット以下になる。６０５０の処理では、

$(A\_R + (-A\_R*p^(-1) \bmod 2^m) * p) / 2^m$ （式１２）

であるが、$A\_R < 2^n$ かつ $(-A\_R * p^(-1) \bmod 2^m) < 2^m$ であるので、

$(A\_R + (-A\_R*p^(-1) \bmod 2^m) * p) / 2^m < 1+p$ （式１３）

となり、必ず６０５０の処理の後の A_R は p以下となり、nビット長で表現できる。また、６０５０でA_Rの値がpと等しくなるのは、xの値がpの倍数となる場合

のみである。

６０３０から６０５０までの処理を１つの式で表現すると、

A_R ≡ x * 2^(n+m) * 2^(m) * 2^(-m) * 2(-m) mod P　　（式１４）

　≡ x * 2^(n+m+m-m-m) mod P　　（式１５）

　≡ x * 2^n mod P　　（式１６）

となる。

６０５０以降の処理は、図３の３０５０以降の処理と同一である。xの値がｐの倍数となっている場合、図２中の２１３０、２１４０の処理にて最終的に補正される。また、図５の実施例と異なり、m < 3n　という条件は必要ない。

【００１４】

図５、図６に示される実施例に必要な、2^L*U mod Pを計算する実施例を図７に示す。図７の手順は下位ビットからのアディションチェインにより、W:=2^L*R mod Pを計算している。Wの初期値は、ｍビットに収まる形で、ｗ≡２＊（２^m） mod Pとなる値にする。Ｌを２進数表現したときに、最上位以外のビットに１がない場合は、L回の剰余自乗計算のみで計算できるが、Ｌ最上位ビット以外に１となるビットが有る場合には、途中で乗算を行う必要がある。従って、Ｌ最上位以外のビット位置に１が見つかったか否かを示す変数mulを用意しておく（７００５）。７０１０の処理は、Ｐの最上位ビットがｍの最上位ビットの位置と等しくなるようにシフトをした値を減じて、ｍビットの値に収まるようにするための処理である。７０２０、７０３０、７０４０、７０５０の処理は、Ｗの最上位２ビットを０とするための処理である。この処理は、最終的な演算結果がｎビットに収まるようにするために行っている。処理が最上位ビットに達した際かのチェックを行い（７０６０）達している場合は、変数mulの値をチェックし（７０８０）、１であれば、ＹにＬの途中のビットに相当する結果が格納されているので、ＷにＹを乗じ（７０９０）結果とする。処理が最上位ビットにまで達していない場合は、Ｌの最下位ビット位置に１があるかチェックを行い（７０７０）見つかった場合は変数ＹにＷの値を保存する。mulの値をチェックし（７１００）、最上位以外のビットで１がはじめて見つかった場合は、ＹにＷの値を代入し（７１２０）、mulに１を設定する（７１３０）。７１２０の処理は、本来Ｙに

１を代入しておきＹ：＝Ｙ＊Ｗと計算することと等しい。また、７１００のチェックで、すでに最上位以外のビットで１が見つかっている場合は、Ｙの値に現在のＷの値を乗じる（７１１０）。Ｒの剰余自乗計算をモンゴメリ剰余乗算で行い（７１４０）、Ｌを１ビット右にシフトし（７１５０）、７０６０からの処理を繰り返す。

７１４０の Y*Y*2^(-m) mod p の計算は、

(A*A+M*p)/(2^m) （式１７）

と等しい。ここで、

M=-Y*Y*(p^(-1))mod 2^m < 2^m （式１８）

である。

したがって、

(Y*Y+M*p)/(2^m) < (Y*Y+2^m*p)/(2^m) （式１９）

また、Ｙをmビット長のメモリに格納したときの、最上位の０となるビットの数をsとすると、

Y<2^(m-s) （式２０）

Y<=2^(m-s)-1 （式２１）

(Y*Y+2^m*p)/(2^m)<=((2^(m-s)-1)^2+2^m*p)/2^m （式２２）

=2^(m-2s)-2^(1-s)+2^(-m)+p （式２３）

<2^(m-2s)+p+1 （式２４）

であるので、７１４０の(Y*Y+M*p)/2^mの演算結果は、１回演算するごとに最上位の0であるビットの数が、s[t+1]:=2s[t]-1となる。従って、2^(m-2s)がpよりも大の場合は、t回計算したあと最上位から連続する０のビット数は、s[0]*2^(t-1)個となり、2^(m-2s)がpよりも小さくなれば、ビット長はｐによって決定される。８０２０、８０３０、８０４０、８０５０の処理で最上位の２ビットは０に設定されるので、s[0]=2となり、t回８１４０を実行した後のビット数は、m-2^t もしくはnとなる。また、tの回数は、t=log2(L)であるので、ビット数は、max(m-L, n)となる。

【００１５】

図８は通常の剰余乗算を用いた場合の本発明の一実施例を示す。ｍを入力ｘの

格納に必要なビット長、ｎをＰの格納に必要なビット長とする。まず、図９のフローに従って、2^n mod Pを計算し、Ｒとする（８０２０）。次に、入力xにＲを乗じる（８０３０）。最後に補正を行うためのR_ITOTALを計算する（８０４０）が、この処理はＰが確定していれば、入力値ｘと独立して計算することが可能であるので、予め計算して保存して置いても良い。実際の計算では、まず2^(-n)mod pを「図１０」の手順に従って計算し、R_INVとし、ＲおよびR_INVを元に図１１の手順に従って、(R_INV)^(2^n-1) mod P を計算し、R_ITOTALとする。通常の剰余乗算を用いるので、演算の初期値Wは１とし、カウンタiに最上位ビットの位置を示すn-1を設定する（８０５０）。指数上位ビットから順に１が立っているかどうかを調べ（８０６０）、１が立っていれば、xR mod P(=A_R) をWに乗じ（４０８０）、ビットの値が０であれば、１を意味するＲを乗じる（８０７０）。ビットの位置を１ビット下に移動し（８０９０）、最下位ビットまで処理が終了したかをチェックし（８１００）、最下位ビットまで処理がたどり着いていない場合は、結果を二乗し（８１１０）、指数の次のビットの処理を行う。８０７０、８０８０の処理では通常の処理と比べると必ずＲが毎回余計に乗じられるため、最下位ビットまで処理が終了した場合は、Ｒ^(2^n-1)が余分に乗じられることになる。余分に乗じられたＲ^(2^n-1)の影響を取り除くために、最後にR_ITOTAL（８２１０）を乗じる。

【００１６】

図９は、図８の実施例中の８０２０で2^L mod Pを計算する処理の一実施例である。図９の手順は下位ビットからのアディションチェインにより、R:=2^L mod Pを計算している。Ｌを２進数表現したときに、最上位以外のビットに１がない場合は、L回の剰余自乗計算のみで計算できるが、Ｌ最上位ビット以外に１となるビットが有る場合には、途中で乗算を行う必要がある。Ｌの最上位以外のビット位置に１が見つかったか否かを示す変数mulを用意し、０に初期化しておく（９００５）。次にＲを２で初期化する（９０１０）。処理が最上位ビットに達した際かのチェックを行い（９０６０）達している場合は、変数mulの値をチェックし（９０８０）、１であれば、ＹにＬの途中のビットに相当する結果が格納されているので、ＲにＹを乗じ（９０９０）結果とする。最上位ビットにまで処理

が達していない場合は、Lの最下位ビット位置に1があるかチェックを行い（9070）見つかった場合は変数Yに保存する。mulの値をチェックし（9100）、最上位以外のビットで1がはじめて見つかった場合は、YにRの値を代入し（9120）、mulに1を設定する（9130）。9120の処理は、本来Yに1を代入しておきY：＝Y＊Rと計算することと等しい。また、9100のチェックで、すでに過去に最上位以外のビットで1が見つかっている場合は、Yの値に現在のRの値を乗じる（9110）。Rの剰余自乗を計算し（9140）、Lを1ビット分右シフトしたのち（9150）、再び9060からの処理に戻る。

【0017】

図10は、図8の実施例中の8040でR_INV:=2^(-n) mod Pを計算する処理の一実施例である。図10の手順は下位ビットからのアディションチェインにより、R_INV:=2^(-L) mod Pを計算している。Lを2進数表現したときに、最上位以外のビットに1がない場合は、L回の剰余自乗計算のみで計算できるが、L最上位ビット以外に1となるビットが有る場合には、途中で乗算を行う必要がある。Lの最上位以外のビット位置に1が見つかったか否かを示す変数mulを用意し、0に初期化しておく（10005）。次にRを1／2で初期化する（10010）。1／2に初期化するには、1を1回右シフトすればよいが、1を右シフトすると0になるため、まずPを加えてから右シフトを行う。Pの値は大きな素数であるので、必ず奇数となる。したがって、1＋Pは必ず偶数になるため、右シフトが可能である。つぎに処理が最上位ビットに達した際かのチェックを行い（10060）達している場合は、変数mulの値をチェックし（10080）、1であれば、YにLの途中のビットに相当する結果が格納されているので、R_INVにYを乗じ（10090）結果とする。最上位ビットにまで処理が達していない場合は、最上位以外のビット位置に1があるかチェックを行い（10070）見つかった場合は変数Yに保存する。mulの値をチェックし（9100）、最上位以外のビットで1がはじめて見つかった場合は、YにRの値を代入し（9120）、mulに1を設定する（9130）。9120の処理は、本来Yに1を代入しておきY：＝Y＊Rと計算することと等しい。また、9100のチェックで、すでに過去に最上位以外のビットで1が見つかっている場合は、Yの値に現在のR

の値を乗じる（９１１０）。Ｒの剰余自乗を計算し（９１４０）、Ｌを１ビット分右シフトしたのち（９１５０）、再び９０６０からの処理に戻る。

【００１８】

図１１は、図８の実施例中の８０４０でR_ITOTAL:=(R_INV)^(2^n-1) mod Pを計算する処理の一実施例である。演算方法は、式２６に示すように、R_INV^(2^n)にRを乗ずることにより行う。R_INV^(2^n)は、剰余自乗演算をｎ回繰り返すことにより計算する。

(R_INV)^(2^n-1) mod P= R_INV^(2^n) * R_INV^(-1) mod P　　　（式２５）

　　　　= R_INV^(2^n) * R mod P　　　（式２６）

まず、R_ITOTALをR_INVで初期化し（１１０１０）、つぎに剰余自乗算を行う回数ｎを変数ｉに代入（１１０２０）、R_ITOTALの剰余自乗した値をR_ITOTALに代入し（１１０３０）、カウンタ用変数ｉから１を減じ（１１０４０）、カウンタ変数ｉの値が0よりも大きければ（１１０５０）、１１０３０からの処理を繰り返す。最後にＲをR_ITOTALに剰余乗算した値をR_ITOTALに代入し（１１０６０）、結果として返す。

付記：

１．法N上での計算において、非演算数にあらかじめNと互いに素な値をべき乗した値を乗じ、計算後に前記Nと互いに素な値のべき乗した値の法Nに対する逆元を乗ずることで計算することを特徴とした処理方法。

２．法N上での剰余計算において、非演算数にあらかじめNと互いに素な値をべき乗した値を乗じ、計算後に前記Nと互いに素な値のべき乗した値の法Nに対する逆元を乗ずることを特徴とした処理方法において、法Nが２よりも大きな素数の積であり、前記Nと互いに素である値として２を用いることを特徴とした処理方法。

３．モンゴメリ剰余乗算装置を有し、Ｐを素数とし、ｘ＞Ｐであるｘに対して、ｘ＊（２＾ｎ）　mod P を計算する情報処理装置に於いて、法をＰ、入力値をｘ、法Ｐを格納するために必要充分なビット数をｎ、入力値ｘを格納するために必要なビット長をｍとし、入力値をｘ＊（２＾ｎ） mod Pに変換する際に、２＾(２m+n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算

装置によりｘ１＝ｘ＊２＾（２ｍ＋ｎ）＊（２＾（−ｍ））mod P＝ｘ＊２＾（ｍ＋ｎ） mod Pを計算し、さらにｘ２：＝ｘ１＊（２＾（−ｍ））mod P＝ｘ＊（２＾ｎ） mod Pを計算することにより、ｘ mod Pを陽に求めること無しにｘ＊（２＾ｎ） mod P を計算することを特徴とする情報処理装置。

４．Ｐを素数とし、ｘ＞Ｐであるｘに対して、ｘ＊（２＾ｎ） mod P を計算する情報処理方法に於いて、法をＰ、入力値をｘ、法Ｐを格納するために必要充分なビット数をｎ、入力値ｘを格納するために必要なビット長をｍとし、入力値をｘ＊（２＾ｎ） mod Pに変換する際に、２＾(２ｍ+n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算によりｘ１＝ｘ＊２＾（２ｍ＋ｎ）＊（２＾（−ｍ））mod P＝ｘ＊２＾（ｍ＋ｎ） mod Pを計算し、さらにｘ２：＝ｘ１＊（２＾（−ｍ））mod P＝ｘ＊（２＾ｎ） mod Pを計算することにより、ｘ mod Pを陽に求めること無しにｘ＊（２＾ｎ） mod P を計算することを特徴とする情報処理方法。

５．モンゴメリ剰余乗算装置を有し、Ｐを素数とし、ｘ＞Ｐであるｘに対して、ｘ＊（２＾ｎ） mod P を計算する情報処理装置に於いて、法をＰ、入力値をｘ、法Ｐを格納するために必要充分なビット数をｎ、入力値ｘを格納するために必要なビット長をｍとし、入力値をｘ＊（２＾ｎ） mod Pに変換する際に、２＾(m+２n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算装置によりｘ１＝ｘ＊２＾（ｍ＋２ｎ）＊（２＾（−ｍ））mod P＝ｘ＊２＾（２ｎ） mod Pを計算し、さらにｘ２：＝ｘ１＊（２＾（−ｎ））mod P＝ｘ＊（２＾ｎ） mod Pを計算することにより、ｘ mod Pを陽に求めること無しにｘ＊（２＾ｎ） mod P を計算することを特徴とする情報処理装置。

６．Ｐを素数とし、ｘ＞Ｐであるｘに対して、ｘ＊（２＾ｎ） mod P を計算する情報処理方法に於いて、法をＰ、入力値をｘ、法Ｐを格納するために必要充分なビット数をｎ、入力値ｘを格納するために必要なビット長をｍとし、入力値をｘ＊（２＾ｎ） mod Pに変換する際に、２＾(m+２n) mod Pを計算するか、あるいは予め用意しておき、モンゴメリ剰余乗算方式によりｘ１＝ｘ＊２＾（ｍ＋２ｎ）＊（２＾（−ｍ））mod P＝ｘ＊２＾（２ｎ） mod Pを計算し、さらにｘ２：＝ｘ１＊（２＾（−ｎ））mod P＝ｘ＊（２＾ｎ） mod Pを計算することによ

り、x mod Pを陽に求めること無しにx＊（2＾n） mod P を計算することを特徴とする情報処理方法。

7．法をP、入力値をx、法Pを格納するために必要充分なビット数をn、入力値xを格納するために必要なビット長をm、指数をdとし、べき乗の指数をsビットづつとりだしてsビット毎のべき乗演算の結果を合成して、x＾d mod Pのべき乗剰余演算を行う情報処理装置において、sビットづつ取り出した指数のi番目の指数をd[i]としたとき、x＾d[i]modPを演算する代わりに（2＾n）＾（2＾s−1）＊x＾d[i]mod Pを用いて演算を行い、（2＾n）＾（2＾n−1）＊x＾d mod Pを計算した後、2＾（−n）＾（2＾n−1） mod Pを乗じて、x＾d mod Pを計算することを特徴とする情報処理装置。

8．法をP、入力値をx、法Pを格納するために必要充分なビット数をn、入力値xを格納するために必要なビット長をm、指数をdとし、べき乗の指数をsビットづつとりだしてsビット毎のべき乗演算の結果を合成して、x＾d mod Pのべき乗剰余演算を行う情報処理方法において、sビットづつ取り出した指数のi番目の指数をd[i]としたとき、x＾d[i]modPを演算する代わりに（2＾n）＾（2＾s−1）＊x＾d[i]mod Pを用いて演算を行い、（2＾n）＾（2＾n−1）＊x＾d mod Pを計算した後、2＾（−n）＾（2＾n−1） mod Pを乗じて、x＾d mod Pを計算することを特徴とする情報処理方法。

【0019】

【発明の効果】

本発明によれば、べき乗剰余演算のCRT計算において、秘密素数による入力値の剰余計算を直接行わずに計算できるため、入力を変えながら、消費電流などを観測することで秘密素数を推定することが困難となる。「図１２」は通例の方法による計算時のx mod Pのビット長およびハミングウエイト（値を２進数表現した場合に１になっているビットの数）を示す。「図１３」は本発明における、x mod Pに相当するx・2＾n mod Pのビット長及びハミングウエイトを示す。図１２では入力データの値とx mod Pの間に明らかな

依存性が現れているが、図13では入力データに依存せず、ビット長およびハミングウエイトが一定値となり、依存性が現れないことが確認できる。

【図面の簡単な説明】

【図1】

通例のRSA用CRT演算方式の処理フロー。

【図2】

モンゴメリ剰余乗算を用いた場合の、通例のCRT方式向けべき乗剰余演算の処理のフロー。

【図3】

xとxを秘密素数pで剰余計算した結果のグラフ。

【図4】

通常の剰余乗算を用いた場合の、通例のCRT方式向けべき乗剰余演算の処理のフロー。

【図5】

本発明による、モンゴメリ剰余乗算を用いたセキュアな剰余演算処理の一実施例。

【図6】

本発明による、モンゴメリ剰余乗算を用いたセキュアな剰余演算処理の別の一実施例。

【図7】

本発明による、モンゴメリ剰余乗算を用いたセキュアな剰余演算処理の部分処理一実施例。

【図8】

本発明による、セキュアなべき乗剰余演算処理の一実施例。

【図9】

本発明による、セキュアなべき乗剰余演算処理の部分処理の一実施例。

【図10】

本発明による、セキュアなべき乗剰余演算処理の部分処理の一実施例。

【図11】

本発明による、セキュアなべき乗剰余演算処理の部分処理の一実施例。

【図１２】

通例のｘとｘを秘密素数ｐで剰余計算した結果のグラフ。

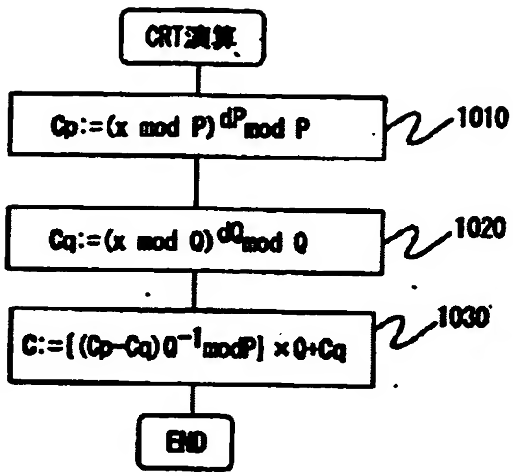
【図１３】

本発明によるｘとｘを秘密素数ｐで剰余計算した結果のグラフ。

【書類名】　図面

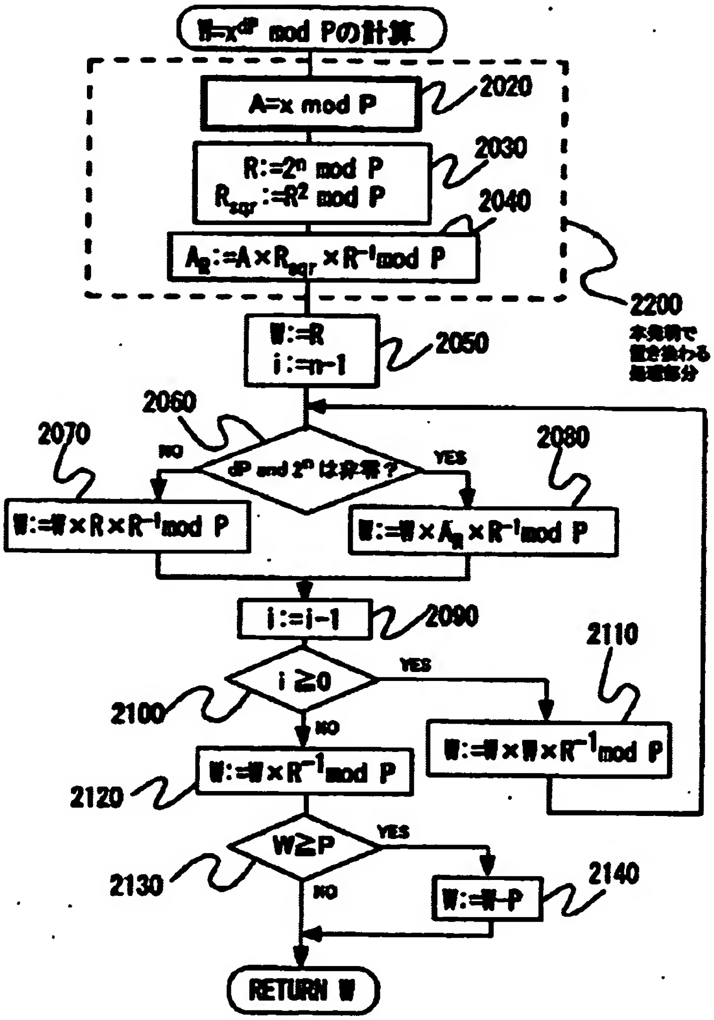
【図1】

図1


CRT演算
Cp:=(x mod P)^dP mod P
1010
Cq:=(x mod Q)^dQ mod Q
1020
C:=[(Cp-Cq)Q^-1 modP]×Q+Cq
1030
END

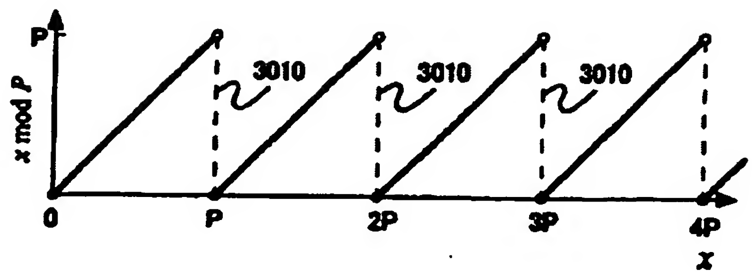
【図２】

図２


W=x^dP mod Pの計算
A=x mod P
2020
R:=2^n mod P
R_sqr:=R^2 mod P
2030
2040
A_R:=A×R_sqr×R^-1 mod P
2200
W:=R
i:=n-1
2050
2060
2070
NO
YES
2080
W:=W×R×R^-1 mod P
W:=W×A_R×R^-1 mod P
i:=i-1
2090
2110
i≧0
YES
2100
NO
W:=W×R^-1 mod P
W:=W×W×R^-1 mod P
2120
W≧P
YES
2130
NO
W:=W-P
2140
RETURN W

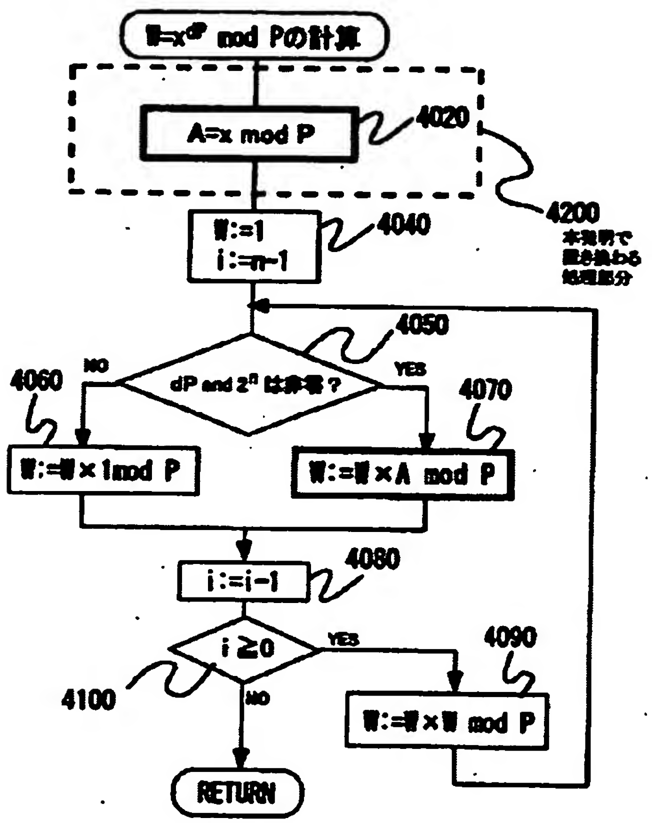
【図3】

図3



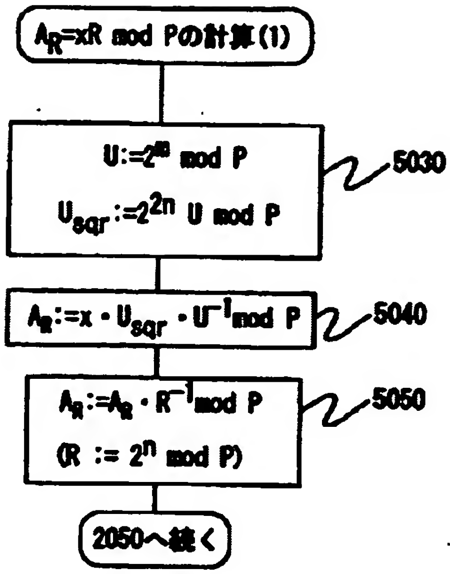
【図4】

図4


W=x^dP mod Pの計算
A=x mod P
4020
4200
本発明で
置き換わる
処理部分
W:=1
i:=n−1
4040
4050
dP and 2^n は非零？
NO
YES
4060
4070
W:=W×1mod P
W:=W×A mod P
i:=i−1
4080
i≧0
YES
NO
4090
4100
W:=W×W mod P
RETURN

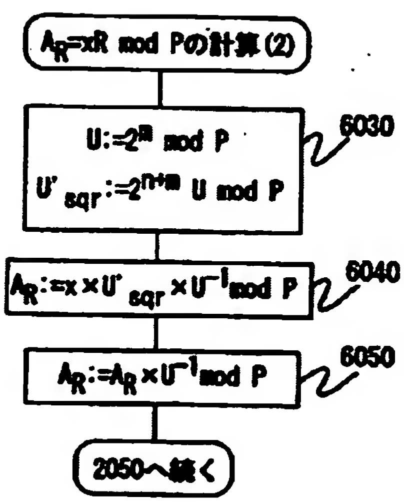
【図5】

図5


A_R=xR mod Pの計算(1)
U:=2^m mod P
U_{sqr}:=2^{2n} U mod P
5030
A_R:=x・U_{sqr}・U^{-1}mod P
5040
A_R:=A_R・R^{-1}mod P
(R := 2^n mod P)
5050
2050へ続く

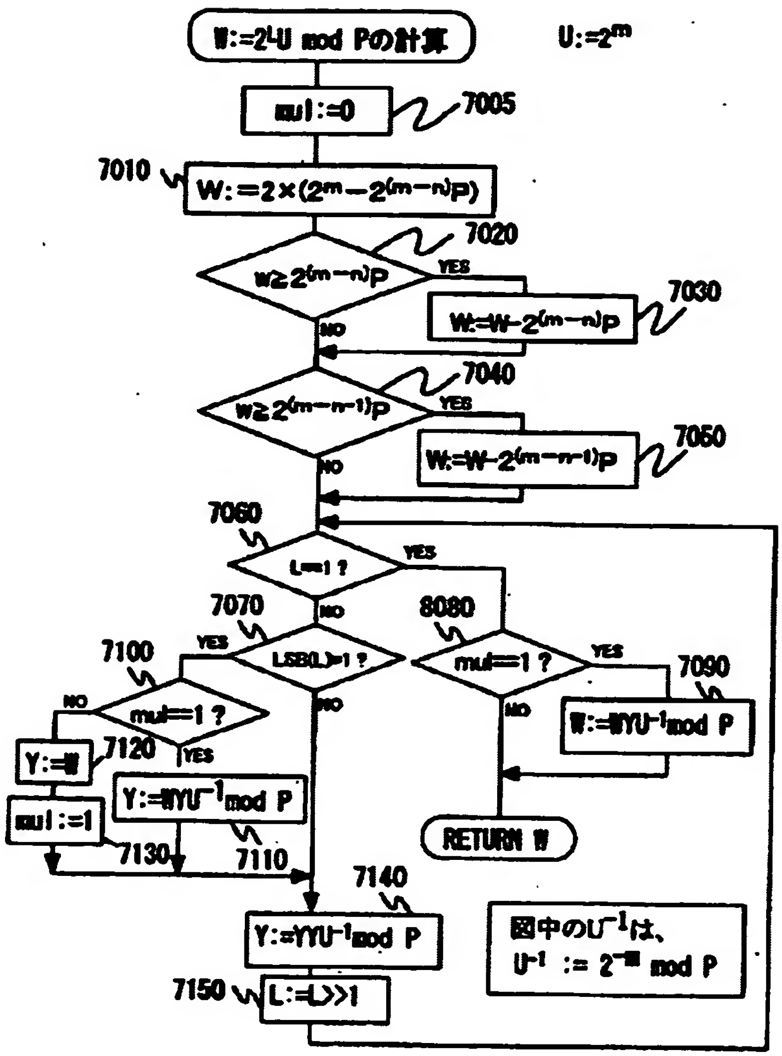
【図6】

図6


A_R=xR mod Pの計算(2)
U:=2^m mod P
U'_{sqr}:=2^{n+m} U mod P
6030
A_R:=x×U'_{sqr}×U^{-1}mod P
6040
A_R:=A_R×U^{-1}mod P
6050
2050へ続く

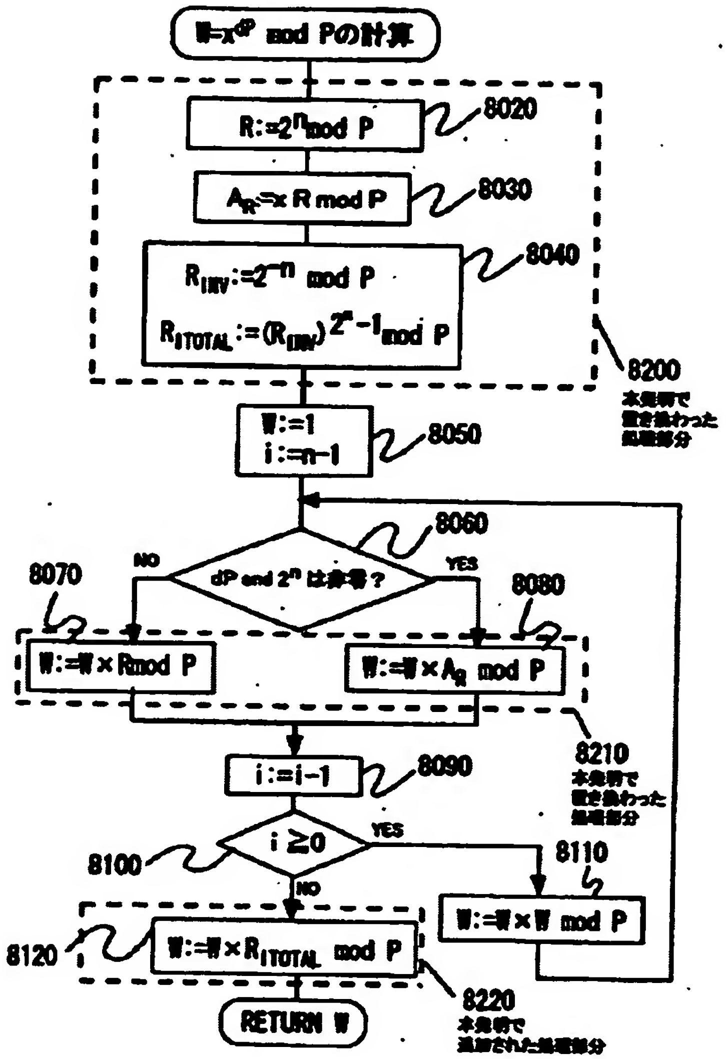
【図7】

図7


W:=2^LU mod Pの計算
U:=2^m
mul:=0
7005
7010
W:=2×(2^m−2^(m−n)p)
7020
W≧2^(m−n)p
YES
NO
7030
W:=W−2^(m−n)p
7040
W≧2^(m−n−1)p
YES
NO
7050
W:=W−2^(m−n−1)p
7060
L==1 ?
YES
NO
7070
LSB(L)==1 ?
8080
mul==1 ?
YES
NO
7090
W:=WYU^-1mod P
7100
mul==1 ?
NO
YES
Y:=W
7120
mul:=1
Y:=WYU^-1mod P
7130
7110
RETURN W
7140
Y:=YYU^-1mod P
図中のU^-1は、
U^-1 := 2^-m mod P
7150
L:=L>>1

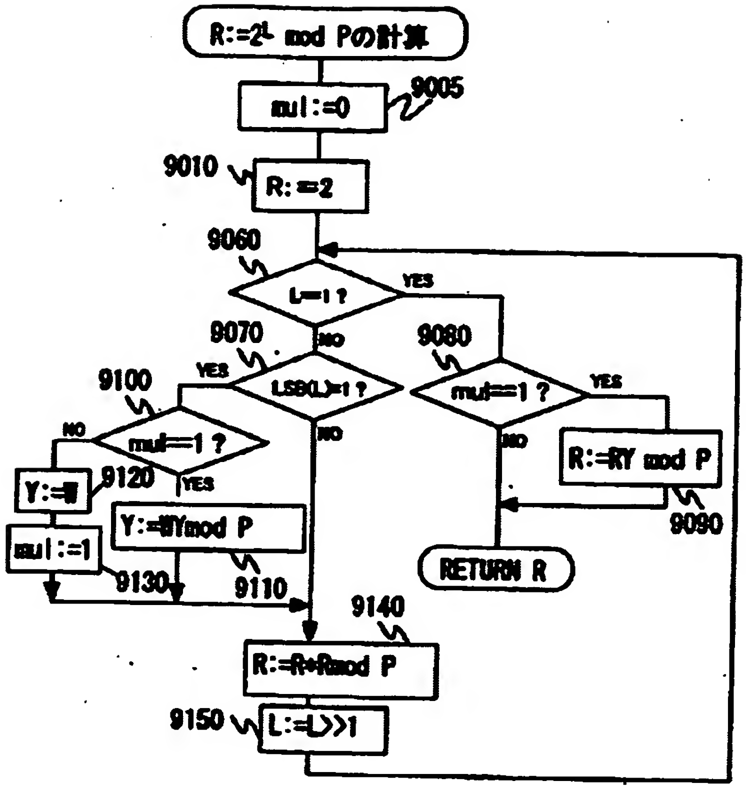
【図８】

図８


W=x^dP mod Pの計算
R:=2^n mod P
8020
A_R:=x R mod P
8030
R_INV:=2^-n mod P
R_ITOTAL:=(R_INV)^(2^n-1) mod P
8040
8200
本発明で
W:=1
i:=n-1
8050
8060
NO
YES
8070
8080
W:=W×R mod P
W:=W×A_R mod P
8210
本発明で
i:=i-1
8090
i≧0
YES
8100
8110
NO
W:=W×W mod P
8120
W:=W×R_ITOTAL mod P
8220
本発明で
RETURN W

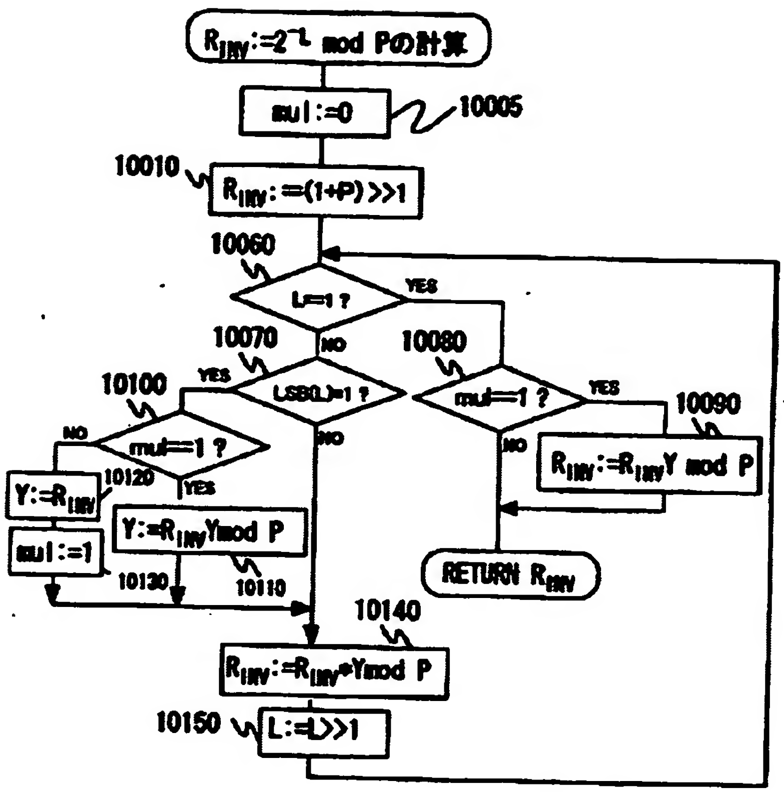
【図9】

図9


R:=2^L mod Pの計算
mul:=0
9005
9010
R:=2
9060
L==1 ?
YES
NO
9070
LSB(L)==1 ?
YES
NO
9080
mul==1 ?
YES
NO
R:=RY mod P
9090
RETURN R
9100
mul==1 ?
NO
YES
Y:=R
9120
Y:=RY mod P
mul:=1
9130
9110
9140
R:=R*R mod P
9150
L:=L>>1

【図10】

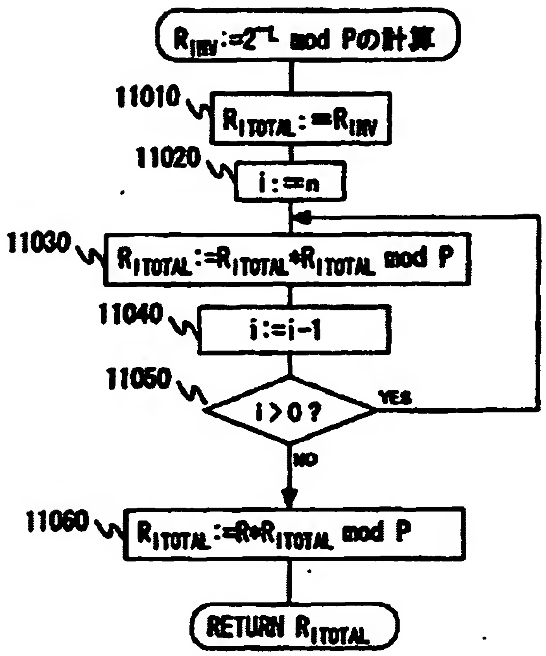
図10



R_INV:=2^-L mod Pの計算
mul:=0
10005
10010
R_INV:=(1+P)>>1
10060
L==1 ?
YES
NO
10070
LSB(L)==1 ?
YES
NO
10080
mul==1 ?
YES
NO
10090
R_INV:=R_INV Y mod P
RETURN R_INV
10100
mul==1 ?
NO
YES
10120
Y:=R_INV
mul:=1
10130
Y:=R_INV Ymod P
10110
10140
R_INV:=R_INV*Ymod P
10150
L:=L>>1

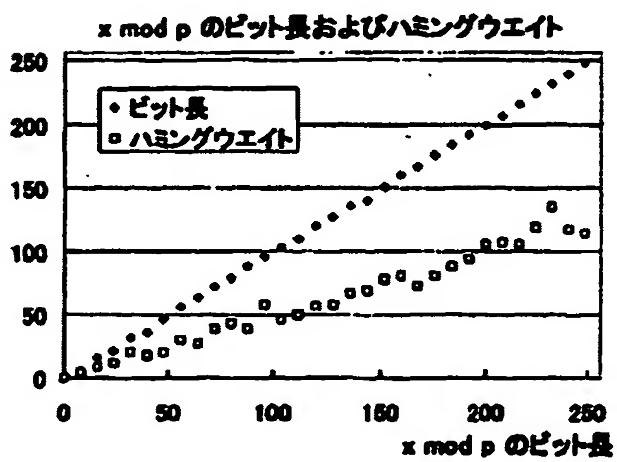
【図11】

図11



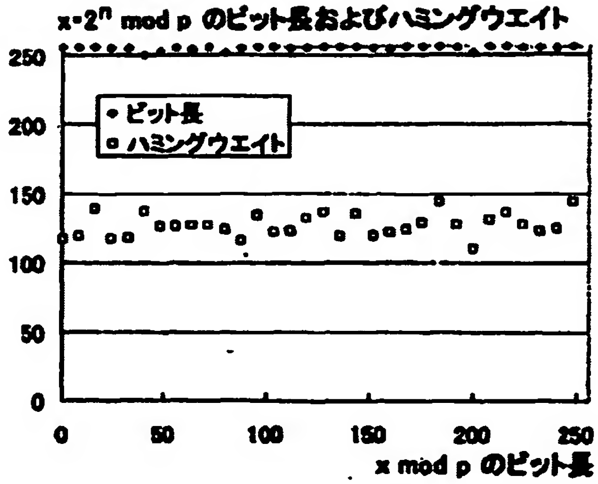
【図12】

図12



【図13】

図13


x·2^n mod p のビット長およびハミングウエイト
250
200
150
100
50
0
0
50
100
150
200
250
x mod p のビット長
• ビット長
□ ハミングウエイト

【書類名】　要約書

【要約】

【課題】　RSA暗号の高速演算手法として、中国人剰余定理を用いた計算手法が広く用いられているが、最初の演算で、秘密素数pによる剰余計算が必要となる。この計算は、秘密素数pを陽に用いた計算であるので、古くからアタックの対象となっている。

【解決手段】　x mod p を直接計算せずに、「図5」に示すように2^(m+n) mod p もしくは、2^(2n) mod pをあらかじめxに乗じておき、その後、2^(-n)もしくは、2^(-m)を乗じて、2^n x mod p を計算する。モンゴメリ剰余乗算を用いる場合は、その後の処理は通例通りとなる。通常の剰余乗算を用いる場合は、べき乗剰余演算の最後に、(2^(-n))^(2^n-1) mod p を乗じ補正する。

【選択図】　図5

認定・付加情報

特許出願の番号	特願2003－014136
受付番号	50300100380
書類名	特許願
担当官	第八担当上席　　　　0097
作成日	平成15年　1月24日

<認定情報・付加情報>

【提出日】　　平成15年 1月23日

次頁無

出 願 人 履 歴 情 報

識別番号　　　　　〔000005108〕

1．変更年月日　　1990年　8月31日
　　［変更理由］　新規登録
　　　住　所　　東京都千代田区神田駿河台4丁目6番地
　　　氏　名　　株式会社日立製作所